# Project

* Site-05-Ng-routing-toast-route-guard-shared-module-misc-components
  + Copied from: Site-04-Ng-login-reg-nav-services-templateforms-Api-MiddlewareException
* For the “MySocialConnect-SPA” issue npm install command to install all packages
  + ng serve : to run the spa

# New & Updates

|  |  |
| --- | --- |
| New | Updates |
| /site/lists | app.routing.module.ts |
| /site/members/member-detail | app.component.html |
| /site/members/member-list | /site/nav.component.html |
| /site/messages | /site/nav.component.ts |
|  | angular.json |
|  | app.module.ts |
|  |  |

# Clear Database

If you want to clear database then drop it and recreate it

dotnet ef database drop

dotnet ef database update

# Creating components

## Components

> ng g c site/registerReactiveform --skip-tests

> ng g c site/members/member-list --skip-tests

> ng g c site/members/member-detail --skip-tests

> ng g c site/members/member-card --skip-tests

> ng g c site/members/member-edit --skip-tests

> ng g c site/members/photo-editor --skip-tests

> ng g c site/lists --skip-tests

> ng g c site/messages --skip-tests

> ng g c site/errors/notLoggedIn --skip-tests

> ng g c site/errors/displayFormGroupErrors --skip-tests

> ng g c site/errors/notFound --skip-tests

> ng g c site/errors/serverError --skip-tests

The components will be added to the app.module.ts as well

## Adding routes to app-routing.module.ts

import { NgModule } from '@angular/core';

import { RouterModule, Routes } from '@angular/router';

import { HomeComponent } from './site/home/home.component';

import { MemberListComponent } from './site/members/member-list/member-list.component';

import { MemberDetailComponent } from './site/members/member-detail/member-detail.component';

import { ListsComponent } from './site/lists/lists.component';

import { MessagesComponent } from './site/messages/messages.component';

const routes: Routes = [

  { path: '', component: HomeComponent },

  { path: 'members', component: MemberListComponent },

  { path: 'members/detail/:guid/:name', component: MemberDetailComponent },

  { path: 'lists', component: ListsComponent },

  { path: 'messages', component: MessagesComponent },

  { path: '\*\*', component: HomeComponent, pathMatch: 'full' } //invalid route

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

# Adding router and Links

## Router Outlet – app.component.html

Remove

<app-home></app-home>

And add in its place

<router-outlet></router-outlet>

## Adding Nav Links – nav.component.html

Title

<a class="navbar-brand" routerLink="/" routerLinkActive="active"><span class="title">{{ title }}</span></a>

Links

        <ul class="navbar-nav me-auto mb-2 mb-md-0" \*ngIf="isLoggedIn">

          <li class="nav-item"><a class="nav-link" routerLink="/members" routerLinkActive="active">Members </a></li>

          <li class="nav-item"><a class="nav-link" routerLink="/lists" routerLinkActive="active">Lists</a></li>

          <li class="nav-item"><a class="nav-link" routerLink="/messages" routerLinkActive="active">Messages</a></li>

        </ul>

Edit profile link

          <div class="dropdown-menu" \*dropdownMenu>

            <a class="dropdown-item" routerLink="members/edit">Edit Profile</a>

            <a class="dropdown-item" (click)="onLogout()">Logout</a>

          </div>

## Routing in code

### nav.componenet.ts

Add the router to constructor

constructor(private helperService: HelperService, private accountService: AccountService,

            private router: Router) { }

On login take the users to he members page

    this.loginSubscription = this.accountService.login(this.loginDto).subscribe({

      next: r => {

        this.helperService.logIfFrom(r, 'Login.User Response');

        //clear the login form

        this.loginDto = <LoginDto>{};

        //route to members

        this.router.navigateByUrl('/members')

      }, error: e => {

        this.helperService.logIfError(e, 'NavComponent.OnLogin');

        this.focusUserName();

      }, complete: () => {

        //set executing login false

        this.isExecutingLogin = false;

      }

    })

  }

On logout take the user to the home page

  onLogout(){

    this.accountService.logout();

    setTimeout(() => {

      this.focusUserName();

      this.router.navigateByUrl("/");

    }, 500);

  }

# Toast Service

## Setup

<https://github.com/scttcper/ngx-toastr>

|  |  |
| --- | --- |
| <https://github.com/scttcper/ngx-toastr>  version 17.0.2 is supports angular 16  > npm install ngx-toastr@17 --save   * Angular animation is needed but it is installed by angular. Check package.json for details * Open angular.json and add the style to styles array 🡺 * Finally add the BrowserAnimationModule and ToastrModule.forRoot() to the imports array in app.module.ts 🡺 | "styles": [   "./node\_modules/ngx-bootstrap/datepicker/bs-datepicker.css",   "./node\_modules/bootstrap/dist/css/bootstrap.min.css",   "./node\_modules/font-awesome/css/font-awesome.min.css",  "./node\_modules/ngx-toastr/toastr.css",  "src/styles.css"              ]  ToastrModule.forRoot({ positionClass: 'toast-bottom-right'}) |

## Use - nav.componenet.ts - login error

Add toastr service

  constructor(private helperService: HelperService, private accountService: AccountService,

            private router: Router,

            private toastr: ToastrService) { }

and then login error

  onLogin(){

    this.helperService.logIfFrom(this.loginDto, 'Login')

    //set executing login true

    this.isExecutingLogin = true;

    //instead of r can also use () or \_

    this.loginSubscription = this.accountService.login(this.loginDto).subscribe({

      next: r => {

        this.helperService.logIfFrom(r, 'Login.User Response');

        //clear the login form

        this.loginDto = <LoginDto>{};

        //route to members

        this.router.navigateByUrl('/members')

      }, error: e => {

        this.helperService.logIfError(e, 'NavComponent.OnLogin');

        this.focusUserName();

        this.toastr.error(e.error);

      }, complete: () => {

        //set executing login false

        this.isExecutingLogin = false;

      }

    })

  }

## Use – register.componenet.ts – registration fail

Inject toastr service

  constructor(private helperService: HelperService, private accountService: AccountService,

    private toastr: ToastrService){}

Update onRegister Method

  onRegister(){

    this.helperService.logIfFrom(this.register, "onRegister TemplateForm");

    this.registerSubscription = this.accountService.register(this.register).subscribe({

      next: () => {

        this.helperService.logIfFrom("", "register template driven");

        this.onCancel();

      },

      error: e => {

        this.helperService.logIfError(e, "register error template driven");

        this.toastr.error(e.error);

      },

      complete: () => {

        this.helperService.logIfFrom("register complete", "register template driven");

      }

    });

  }

# Route Guard

## /core/guards/auth.guard.ts - CanActivate

> ng g g core/guards/auth --skip-tests

? Which type of guard would you like to create? CanActivate

import { inject } from '@angular/core';

import { ActivatedRouteSnapshot, CanActivateFn, Router, RouterStateSnapshot } from '@angular/router';

import { AccountService } from '../services/account.service';

import { ToastrService } from 'ngx-toastr';

import { map } from 'rxjs';

import { LoggedInUserDto } from '../models-interfaces/logged-in-user-dto.model';

export const authGuard: CanActivateFn = (route: ActivatedRouteSnapshot, state: RouterStateSnapshot) => {

  const accountService = inject(AccountService);

  const toastr = inject(ToastrService);

  const router = inject(Router);

  return accountService.currentLoggedInUser$.pipe(

    map(user => {

      if(user)

        return true;

      else{

        //in real time we do not do this.

        toastr.error("You are not authorized to view the resource. Login please!");

        //redirect to the error page

        router.navigate(["/errors/notloggedin"], {queryParams: { returnUrl: state.url } });

        return false;

      }

    })

  );

};

## Apply guard to the routes – app-routing.module.ts

### Approach 1

  { path: 'members', component: MemberListComponent, canActivate: [authGuard] },

  { path: 'members/detail/:guid/:name', component: MemberDetailComponent, canActivate: [authGuard] },

  { path: 'members/edit', component: MemberEditComponent, canActivate: [authGuard] },

  { path: 'lists', component: ListsComponent, canActivate: [authGuard] },

  { path: 'messages', component: MessagesComponent, canActivate: [authGuard] },

### Approach 2 – Dummy route

const routes: Routes = [

  { path: '', component: HomeComponent },

  //dummy route to group secure resources together

  {

    path: '',

    runGuardsAndResolvers: 'always',

    canActivate: [authGuard],

    children: [

      { path: 'members', component: MemberListComponent },

      { path: 'members/detail/:guid/:name', component: MemberDetailComponent },

      { path: 'members/edit', component: MemberEditComponent },

      { path: 'lists', component: ListsComponent },

      { path: 'messages', component: MessagesComponent },

    ]

  },

  { path: 'errors/notloggedin', component: NotLoggedInComponent },

  { path: '\*\*', component: HomeComponent, pathMatch: 'full' } //invalid route

];

## /site/errors/not-logged-in

### not-logged-in.component.ts

import { Component, OnDestroy, OnInit } from '@angular/core';

import { ActivatedRoute, ParamMap, Router } from '@angular/router';

import { Subscription, combineLatest, forkJoin } from 'rxjs';

import { AccountService } from '../../../core/services/account.service';

import { HelperService } from '../../../core/services/helper.service';

@Component({

  selector: 'app-not-logged-in',

  templateUrl: './not-logged-in.component.html',

  styleUrls: ['./not-logged-in.component.css']

})

export class NotLoggedInComponent implements OnInit, OnDestroy {

  returnUrl: string = "";

  isLoggedIn: boolean = false;

  querySubscription!: Subscription;

  loggedInUserSubscription!: Subscription;

  forkjoinSubscription!: Subscription;

  combineLatestSubscription!: Subscription;

  constructor(private activateRoute: ActivatedRoute,

            private accountService: AccountService,

            private helperService: HelperService,

            private router: Router){}

  ngOnDestroy(): void {

    if(this.querySubscription) this.querySubscription.unsubscribe();

    if(this.loggedInUserSubscription) this.loggedInUserSubscription.unsubscribe();

    if(this.forkjoinSubscription) this.forkjoinSubscription.unsubscribe();

    if(this.combineLatestSubscription) this.combineLatestSubscription.unsubscribe();

  }

  ngOnInit(): void {

    //this.individual();

    this.together();

  }

  individual(){

    this.loggedInUserSubscription = this.accountService.currentLoggedInUser$.subscribe({

      next: r => {

        this.isLoggedIn = !!r;

        this.getParams();

      },

      error: e => {

        this.helperService.logIfError(e, "getCurrentLoggedInUser")

      }

    });

  }

  getParams(){

    //one way of getting the query param

    /\*

    const returnUrl = this.activateRoute.snapshot.queryParamMap.get("returnUrl");

    if(returnUrl)

      this.returnUrl = returnUrl;

    \*/

    //second way of getting the query param - use this

    this.querySubscription = this.activateRoute.queryParamMap.subscribe({

      next: (params: ParamMap) => {

        const returnUrl = params.get("returnUrl");

        if(returnUrl)

          this.returnUrl = returnUrl;

        if(this.isLoggedIn){

          if(this.returnUrl)

            this.router.navigateByUrl(this.returnUrl);

          else

            this.router.navigateByUrl('/members');

        }

      }

    });

  }

  together(){

    const user$ = this.accountService.currentLoggedInUser$;

    const param$ = this.activateRoute.queryParamMap;

    /\*

    forkjoin the subscription must end

    this.forkjoinSubscription = forkJoin([user$, param$]).subscribe({

      next: result => {

        this.doAction(result);

      },

      error: e => {

        console.log(e);

      },

      complete: () => {

        console.log('complete');

      }

    });

    \*/

    this.combineLatestSubscription = combineLatest([user$,param$]).subscribe({

      next: r => {

        this.doAction(r);

      }

    });

  }

  doAction(r: any){

    const userResult = r[0];

    const paramResult = r[1];

    this.isLoggedIn = !!userResult;

    const temp = paramResult.get('returnUrl');

    if(temp) this.returnUrl = temp;

    if(this.isLoggedIn){

      let redirectTo = "/members";

      if(temp)

        redirectTo = temp;

      this.router.navigateByUrl(redirectTo);

    }

  }

}

### not-logged-in.component.html

<h2>Unauthorized</h2>

<p>You have accessed a secure page. Please login at the top of this page.</p>

<p \*ngIf="returnUrl">URL: {{ returnUrl }}</p>

## /site/nav/nav.component.ts

On login redirect the users to the returnUrl if available otherwise to the /members page

Inject activate route

private activatedRoute: ActivatedRoute

Setup variables

  returnUrl: string = "";

  queryParamsubscription!: Subscription;

un subscribe the subscription

  ngOnDestroy(): void {

    if(this.loginSubscription) this.loginSubscription.unsubscribe();

    if(this.loggedInUserSubscription) this.loggedInUserSubscription.unsubscribe();

    if(this.queryParamsubscription) this.queryParamsubscription.unsubscribe();

  }

on ngOnInit get the param

    //fill return url if avaialble

    //method #1

    //const returnUrl1 = this.activatedRoute.snapshot.queryParamMap.get('returnUrl');

    //if(returnUrl1)

    //  this.returnUrl = returnUrl1;

    //method #2

    this.queryParamsubscription = this.activatedRoute.queryParamMap.subscribe({

      next: (params: ParamMap) => {

        const returnUrl = params.get("returnUrl");

        if(returnUrl)

          this.returnUrl = returnUrl;

      }

    });

onLogin redirect the user accordingly

        //route to members

        const returnUrl = this.returnUrl ? this.returnUrl : '/members';

        this.router.navigateByUrl(returnUrl);

# Theming

## Boostwatch

<https://bootswatch.com/united/>

> npm install bootswatch

Add the theme unit css to angular.json

"styles": [

              "./node\_modules/ngx-bootstrap/datepicker/bs-datepicker.css",

              "./node\_modules/bootstrap/dist/css/bootstrap.min.css",

              "./node\_modules/bootswatch/dist/spacelab/bootstrap.min.css",

              "./node\_modules/font-awesome/css/font-awesome.min.css",

              "./node\_modules/ngx-toastr/toastr.css",

              "src/styles.css"

            ]

## nav.component.html

remove bg-dark and add bg-primary

<nav class="navbar navbar-expand-md navbar-dark fixed-top bg-primary"> <!-- bg-dark-->

# Shared Module

> ng g m core/modules/shared –flat

|  |  |
| --- | --- |
| app.module.ts | core/modules/shared.module.ts |
| import { NgModule } from '@angular/core';  import { BrowserModule } from '@angular/platform-browser';  import { SharedModule } from './core/modules/shared.module';  import { AppComponent } from './app.component';  import { NavComponent } from './site/nav/nav.component';  import { NotFoundComponent } from './site/errors/not-found/not-found.component';  import { ServerErrorComponent } from './site/errors/server-error/server-error.component';  @NgModule({    declarations: [      AppComponent,      NavComponent,      NotFoundComponent,      ServerErrorComponent    ],    imports: [      BrowserModule,      SharedModule,    ],    providers: [],    bootstrap: [AppComponent]  })  export class AppModule { } | import { NgModule } from '@angular/core';  import { CommonModule } from '@angular/common';  import { BrowserAnimationsModule } from '@angular/platform-browser/animations';  import { HttpClientModule } from '@angular/common/http';  import { FormsModule, ReactiveFormsModule } from '@angular/forms';  import { BsDropdownModule } from 'ngx-bootstrap/dropdown';  import { ToastrModule } from 'ngx-toastr';  //app routing module  import { AppRoutingModule } from '../../app-routing.module';  //directive for using the validator in template driven forms  import { MustMatchDirective } from '../directives/must-match.directive';  import { PasswordStrengthDirective } from '../directives/password-strength.directive';  import { UserNameAllowedDirective } from '../directives/user-name-allowed.directive';  import { UserNameCheckDirective } from '../directives/user-name-check.directive';  //components  import { HomeComponent } from '../../site/home/home.component';  import { RegisterComponent } from '../../site/register/register.component';  import { MemberListComponent } from '../../site/members/member-list/member-list.component';  import { MemberDetailComponent } from '../../site/members/member-detail/member-detail.component';  import { ListsComponent } from '../../site/lists/lists.component';  import { MessagesComponent } from '../../site/messages/messages.component';  import { MemberCardComponent } from '../../site/members/member-card/member-card.component';  import { MemberEditComponent } from '../../site/members/member-edit/member-edit.component';  import { PhotoEditorComponent } from '../../site/members/photo-editor/photo-editor.component';  import { NotLoggedInComponent } from '../../site/errors/not-logged-in/not-logged-in.component';  import { DisplayFormGroupErrorsComponent } from '../../site/errors/display-form-group-errors/display-form-group-errors.component';  import { RegisterReactiveformComponent } from '../../site/register-reactiveform/register-reactiveform.component';  @NgModule({    declarations: [      MustMatchDirective, //for validation template driven form      PasswordStrengthDirective, //for validation template driven form      UserNameAllowedDirective, //for validation template driven form      UserNameCheckDirective, //for validation template driven form      HomeComponent,      RegisterComponent, //template driven form      RegisterReactiveformComponent, //reactive form      MemberListComponent,      MemberDetailComponent,      ListsComponent,      MessagesComponent,      MemberCardComponent,      MemberEditComponent,      PhotoEditorComponent,      NotLoggedInComponent,      DisplayFormGroupErrorsComponent,    ],    imports: [      CommonModule,      AppRoutingModule,      BrowserAnimationsModule,      HttpClientModule,      FormsModule,      ReactiveFormsModule,      BsDropdownModule.forRoot(),      ToastrModule.forRoot({ positionClass: 'toast-bottom-right'}),    ],    exports: [      AppRoutingModule,      BrowserAnimationsModule,      HttpClientModule,      FormsModule,      ReactiveFormsModule,      BsDropdownModule,      ToastrModule,    ]  })  export class SharedModule { } |